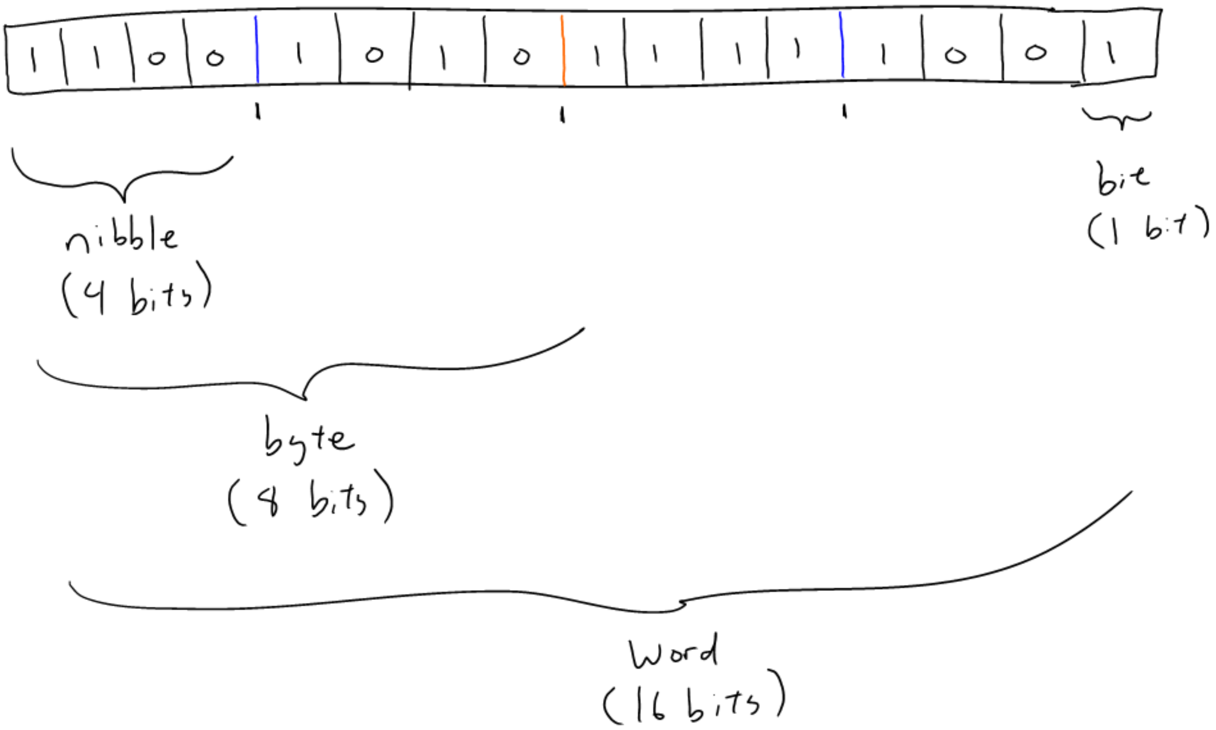
Super Simple Number Theory : HEX, DEC, BCD

R1.2

Remember that the int datatype on this platform is 16 bits.

That is 16-bits, or 2 bytes, or 4 nibbles:



You must remain aware of how big things are when programming, so that you don't lose bits with your expressions.

For example, assignment of 16 bits to an 8-bit type will truncate, and lose the upper 8 bits:

```
unsigned int  iStuff = 0x1234;
unsignd char  cStuff = iStuff;  // loss of data!
```

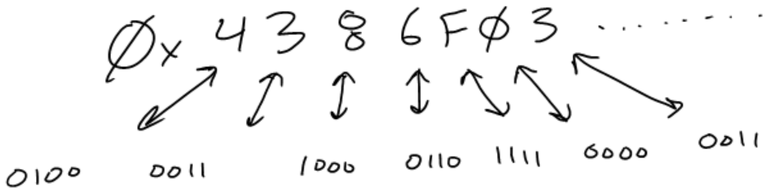
cStuff will contain 0x34 (the lower byte).

Sometimes you may want this, but it should always be cast.

```
cStuff = (unsigned char) iStuff;
```

Numbers are stored as bits. Sign and base are interpretations we apply. So when you code x = 0x4386; or x = 17286, this is for YOU, not the compiler.

HEX is a useful representation, as it can easily be bidirectionally converted to binary.



You must become good @ converting HEX ⇄ BIN.

Use the correct base in code for context.

```
unsigned char letter = 'A';  ← This is a number!
unsigned int iNumPeople = 22;
unsigned char cMask = 0b00110011;
unsigned int iMaxVal = 0xFFFF;
```

BCD

When we show numbers to humans on displays, they tend to like them in decimal. It would be odd if your microwave used HEX. This is what our 7-segs do by default. In fact, HEX/BIN will be very common in hardware.

If you want to show a number to the user, you can still show it in HEX, but make it look like decimal!

This is BCD, or "binary-coded decimal". This is for display only, and has no other purpose.

Since a number is a number in code, you are free to apply base interpretation how you like. If we take a number and use %10, we will isolate the least value decimal position. If you divide by 10, the number will shift right one digit (just like shift >>1 in binary divides by 2).

Using divide and modulus w/ base radix isolates digits!

↓

10 in the case of decimal.

Example: Isolate the digits of 4096_{10}

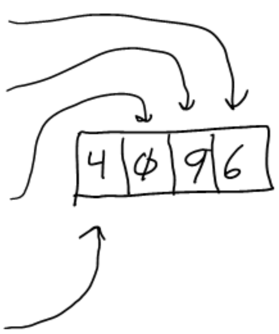
The source number is just a number!

$$(\underline{4096} / 1) \% 10 = 6$$

$$(\underline{409}6 / 10) \% 10 = 9$$

$$(\underline{40}96 / 100) \% 10 = 0$$

$$(\underline{4}096 / 1000) \% 10 = 4$$



You could put these individual digits out to the 7-segs using the Segs_Normal function. The function "thinks" they are HEX (or could be HEX), but they are in the decimal range. A HEX number that looks like decimal (BCD).

This would only work for numbers in the range 0 → 9999, as we only have 4 display elements per line.

The Segs_16D (unsigned int Value, unsigned char Line) function does this.

0 → 9999

0 or 1
Top Bottom

If the user passes a value out of range, what do you do?

Discuss, and agree on implementation!

Segs_16H() can shift to isolate nibbles, w/AND.

0x1234 & 0x000F = 4

>>= 4

0x0123 & 0x000F = 3

⋮

But it could do the same trick w/ radix 16:

$0x4A / \underline{0}x10 = 4$

$0x4A \% \underline{0}x 10 = A$